SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Marko Cundeković

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Mentor:
Prof.dr.sc. Mario Essert

Student:
Marko Cundeković

Zagreb, 2013.

Izjavljujem da sam ovaj rad izradio samostalno koristeći stečena znanja tijekom studija i navedenu literaturu.

Ovom prilikom bih se želio zahvaliti:

Voditelju rada Prof.dr.sc. Mariu Essertu što mi je omogućio izradu ovog rada, te na sugestijama i pomoći pri izradi istog.

Obitelji i prijateljima na pomoći i potpori, kako za ovaj rad, tako i kroz sve godine studiranja.

Marko Cundeković

SVEUČILIŠTE U ZAGREBU

FAKULTET STROJARSTVA I BRODOGRADNJE

Središnje povjerenstvo za završne i diplomske ispite

Povjerenstvo za završne ispite studija strojarstva za smjerove:
proizvodno inženjerstvo, računalno inženjerstvo, industrijsko inženjerstvo i menadžment, inženjerstvo materijala i mehatronika i robotika

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum	Prilog
Klasa:	
Ur.broj:	

ZAVRŠNI ZADATAK

Student: **MARKO CUNDEKOVIĆ** Mat. br.: 0035157307

Naslov rada na hrvatskom jeziku: **Povezani podaci u novom ustroju Interneta**

Naslov rada na engleskom jeziku: **Linked Data as New Internet Framework**

Opis zadatka:

Dosadašnja slika Interneta je povezanost WEB stranica preko oznaka (<href> tag) koje WEB preglednik (browser) razumije i izvodi. U starom ustroju teško je pretraživati raspoloživu informaciju, jer je broj nađenih stranica ('pogodaka') preko Google pretraživača najčešće veći od više stotina tisuća. Rješenje se nazire uvođenjem novog, paralelnog ustroja – semantičkog weba i to zamjenom poveznica na web-stranice s poveznicama na podatke (URI data link).

U ovo radu potrebno je istražiti i provesti:

1. Osnove 'Linked data' paradigme.
2. Temeljnu strukturu podataka u obliku RDF trojaca (RDF triplet) – subjekta, objekta i predikata kao osnovnog čvora WEB grafa.
3. U PHP/MySQL programu načiniti bazu RDF trojaca u kojoj će se upisivati i mijenjati podaci iz XML datoteka ili preko ulaznog korisnikovog sučelja, (HTML/CSS) obrasca.
4. Opisati SPARQL bazu i jezik za nova traženja umreženih podataka.
5. Predstaviti IsaViZ program za vizualizaciju RDF podataka.

Zadatak zadan:

16. studenog 2012.

Zadatak zadao:

Prof.dr.sc. Mario Essert

Rok predaje rada:
1. rok: 15. veljače 2013.
2. rok: 11. srpnja 2013.
3. rok: 13. rujna 2013.

Predviđeni datumi obrane:
1. rok: 27., 28. veljače i 1. ožujka 2013.
2. rok: 15., 16. i 17. srpnja 2013.
3. rok: 18., 19., i 20. rujna 2013.

Predsjednik Povjerenstva:

Prof. dr. sc. Zoran Kunica

SADRŽAJ

POPIS SLIKA

SAŽETAK

U ovome radu će biti opisana ideja i tehnologije koje se koriste za izradu semantičkog weba, odnosno način povezivanja podataka na web-u kako bi se stvorila globalna mreža znanja. Temeljna ideja semantičkog web-a je davanje značenja podacima na način da ih računalo razumije. Objašnjen je osnovni model RDF (Resource Description Framework) te tehnologije i jezici na koje se on oslanja. U drugom dijelu rada je izrađena stranica pomoću PHP-a, MySQL-a i ARC2 sustava koja prikazuje korištenje povezanih podataka u praksi. Na njoj je moguće upisivati RDF trojce u bazu podataka iz datoteka, te pretraživati bazu preko SPARQL jezika. Za kraj je predstavljen IsaViz program kojim možemo RDF trojce stvarati, editirati i prikazivati preko grafičkog sučelja.

Ključne riječi: semantički web, povezani podaci, XML, RDF, OWL, SPARQL, ARC2

1. UVOD

Godine 1989., Tim Berners – Lee, današnji predsjednik World Wide Web Consortium-a (W3C), predstavio je osnovnu ideju za današnji internet. Kako je i sam izjavio[1], želio je povezati računala u zajedničku mrežu kako bi olakšao dijeljenje podataka i informacija među kolegama. Tada je korištenjem HTTP-a (HyperText Transfer Protocol) pokrenut World Wide Web, a 1991. godine javnosti je predstavljena prva internet stranica.

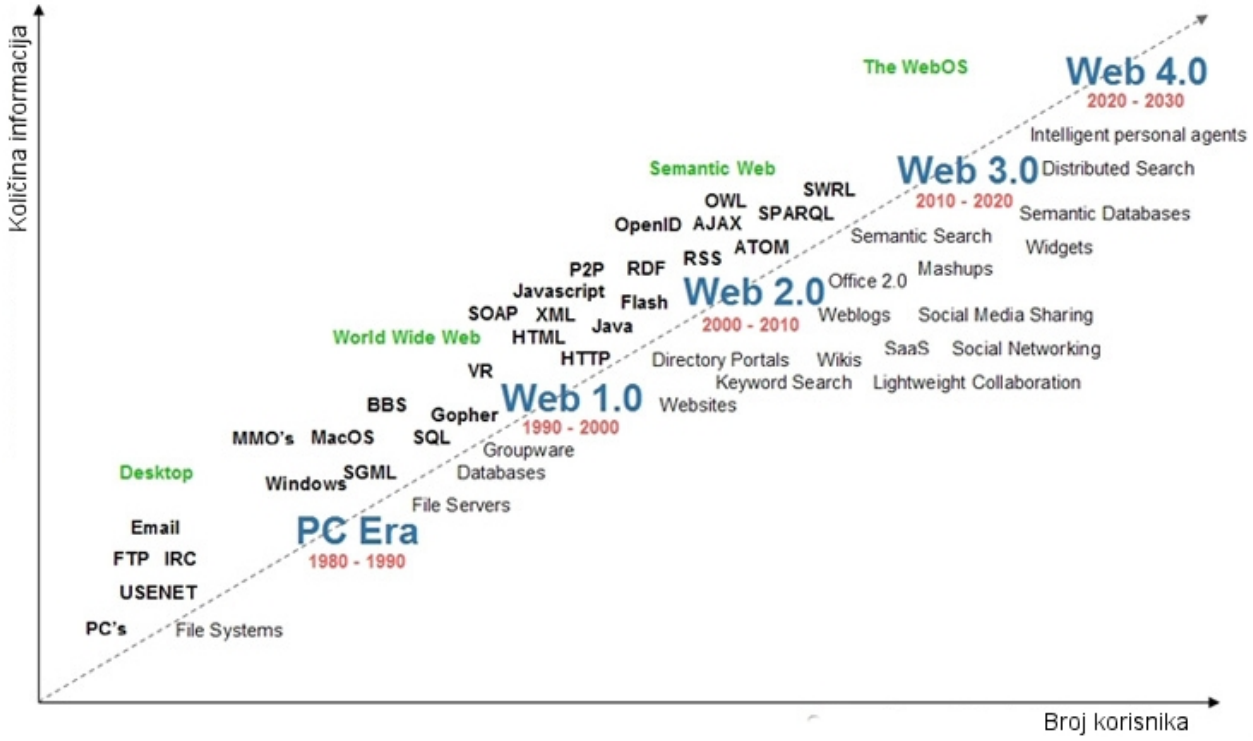
S razvojem weba te sve većim brojem informacija i datoteka na njemu, došlo je do problema kod pronalaženja istih. Danas je sve teže doći do željene informacije bez da nam se ne ponudi velik broj nevezanih za našu pretragu. Današnji pretraživači rade na principu podudaranja traženih ključnih riječi, tako da pri pretraživanju mi pretražujemo da li se naš traženi izraz pojavljuje na nekoj stranici, a kao rezultat dobivamo adresu tih stranica i sami provjeravamo da li tamo postoji željeni odgovor.

Uzmimo za primjer Google na kojem želimo pretražiti informacije o semantičkom webu sa ključnim riječima *„semantic web“*. Trenutni rezultati pretraživanja nam daju 16,500,000 adresa web stranica na kojima se taj izraz spominje. Mi i dalje ne znamo kakve informacije nam te stranice nude, jesu li one točne i jesu li uopće povezane sa našim upitom. Što se web više nadopunjava i širi, nama postaje sve teže doći do relevantnih i točnih podataka.

„Web often feels like it is a mile wide but an inch deep.” [2]

Također postoji problem jezika na kojem su informacije napisane. Iako automatski prijevodi stranica ili ključnih riječi znaju nekada biti od koristi, dosta često se kod takvog računalnog prevođenja izgubi smisao rečenice. Isto tako postoji problem pretrage homonima. S obzirom da velika i mala slova ne utječu na osnovno pretraživanje, riječ *„bor“* može predstavljati vrstu drveta, kemijski element, ime grada, ime plemena i ime lika iz nordijske mitologije. U tom slučaju algoritam ne može razaznati na što korisnik misli te prikazuje rezultate vezane za sve pojmove.

Web s vremenom, osim povećanja količine podataka proširuje i svoju namjenu. Slika 1. prikazuje razvoj od ere prije interneta do (skorije) budućnosti.



Slika 1. - Razvoj web-a [3]

Nakon pojave World Wide Web-a određen je i standard Web 1.0. U tom vremenu su internet stranice bile statične. Iako je komunikacija između računala u tehničkom smislu bila dvosmjerna, krajnji korisnici su samo mogli prikupljati informacije i datoteke sa web-a, dok su autori stranica bili oni koji su informacije postavljali na njega.

S daljnjim razvojem, nastao je Web 2.0 standard. Došlo je do značajne promjene kojoj je glavno obilježje dinamičnost. Svi korisnici su dobili priliku sudjelovati u stvaranju i kreiranju sadržaja web stranica bez potrebnog dubljeg znanja programiranja, na stranicama poput Wikipedije. Web je počeo služiti i za komunikaciju između krajnjih korisnika. Kombinacijom toga, pojavile su se nezaobilazne društvene mreže koje su pravi primjeri Web 2.0. Facebook, Twitter, MySpace su neke od njih na kojima uz potpuno dinamično korisničko sučelje, korisnici sami stvaraju sadržaj tih stranica i međusobno komuniciraju, dok se one u osnovi sastoje samo od jezgre stranice bez značajnijeg sadržaja.

Danas smo na prijelazu Web 2.0 na Web 3.0 standard koji također donosi veliki korak naprijed u funkcionalnosti web-a. Zbog prethodno navedenih problema, cilj je stvoriti semantički web, odnosno povezati podatke i dati značenje njihovoj povezanosti koje će i računalo razumjeti što će ujedno biti i glavno obilježje novog standarda.

1.1. Semantički Web

Kako bi se pokušali riješiti prethodno navedeni problemi današnjeg weba, Tim Berners – Lee je 2001. godine predstavio koncept modela semantičkog weba.

„Semantika (grčki semantikos, koji daje znakove, značajan, simptomatičan, od sema, znak) se odnosi na aspekte značenja koji su izraženi u jeziku, kodu ili nekom drugom obliku predstavljanja.“ [4]

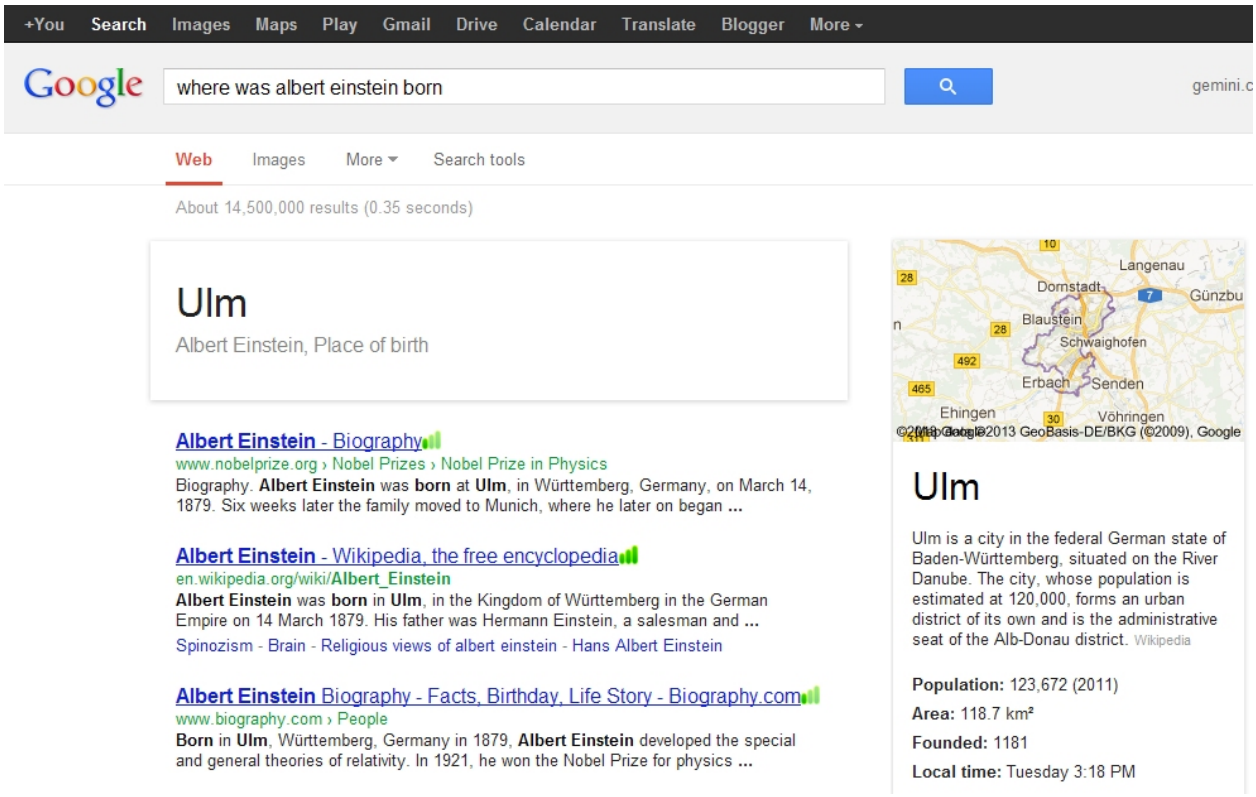
Kao što se vidi iz definicije, prva ideja iza ovog modela je davanje značenja i definiranje podataka kako bi korisnici, u svakodnevno sve većem broju novih, lakše pronalazili i filtrirali željene informacije. Druga ideja semantičkog weba je omogućiti samim računalima da razumiju naše upite na način kako ih i ljudi shvaćaju te nam na taj način olakšaju pretragu i povežu podatke.

Uzmimo jedan primjer pretraživanja bez takvog modela i sa njim. Kada bismo željeli saznati mjesto rođenja Alberta Einsteina, u tražilicu možemo upisati *„Where was Albert Einstein born?“*. Kao rezultat bi dobili samo adrese stranica koje sadržavaju ključne riječi a to su „*albert*“, „*einstein*“ i „*born*“. Takvih stranica trenutno postoji 14,500,000. Nakon toga otvaramo stranicu po stranicu kako bi pronašli (ako postoji) podatak o mjestu rođenja. Iako mi razumijemo što smo tražili i razumijemo nađene rezultate, za računalo to predstavlja samo niz znakova koji su se podudarali. Same riječi nemaju nikakvo smisleno značenje niti postoji semantička veza između njih.

Sa implementiranim modelom semantičkog web-a, na isto pitanje, kao rezultat bi dobili jedan jedini, a taj je *„Ulm“*, što je ustvari naš željeni odgovor. Uz to, računalo bi razumjelo da tražimo mjesto, znalo bi gdje se to mjesto nalazi, kolika mu je površina, koliko stanovnika ima i ostale informacije. Računalu je točno definiran podatak o kojem se radi.

Iako je ideja semantičkog web-a stara preko 10 godina te nakon prvog objavljivanja nije zaživjela u tolikoj mjeri kao ostali segmenti Web 2.0 standarda, zbog brzog širenja interneta i ogromnih količina podataka, vodeći internet servisi poput Google-a, Facebook-a i mnogih drugih, danas je implementiraju uz postojeću tehnologiju.

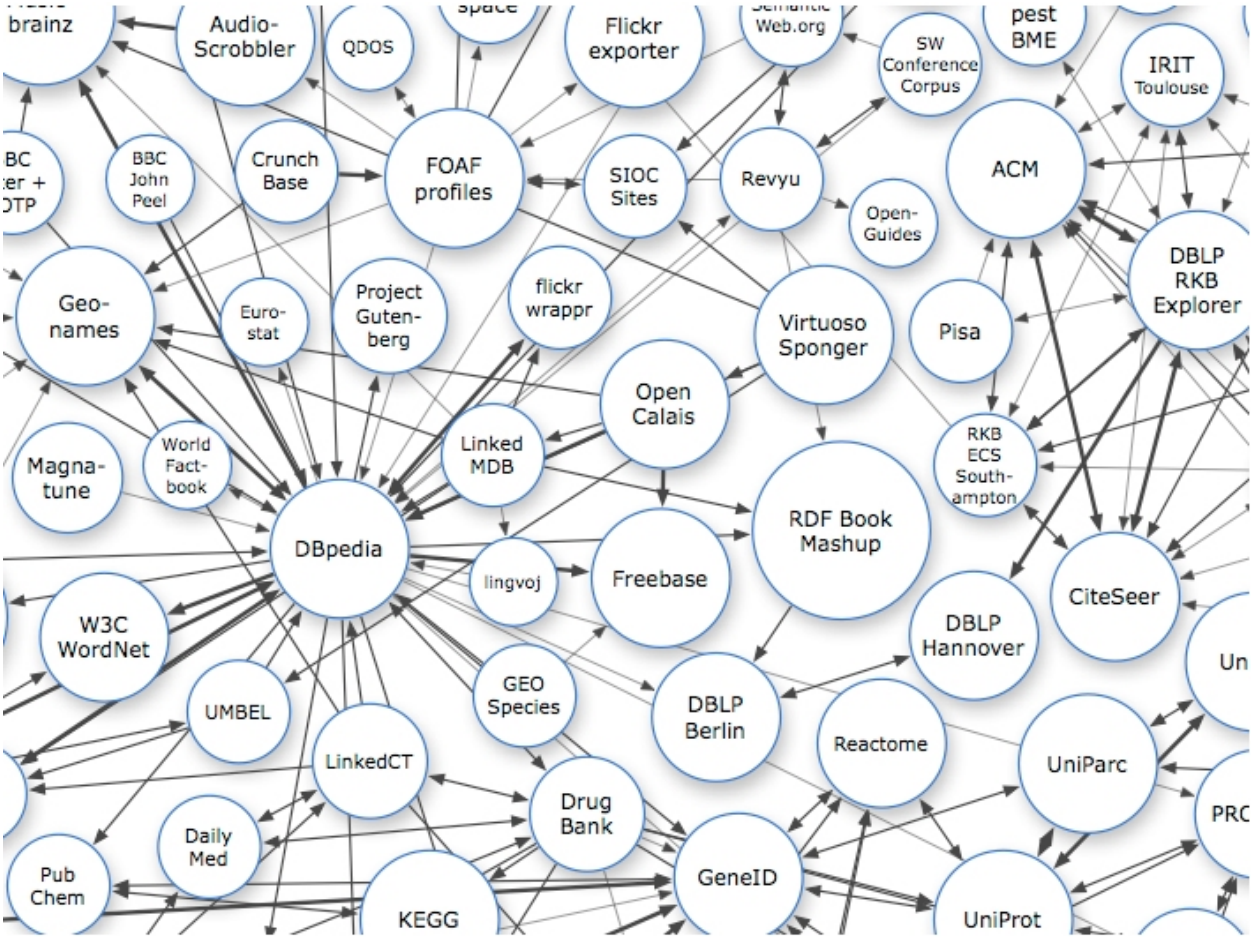
Google je predstavio *Google knowledge graph*. Uz dosadašnji način pretraživanja podataka, nadogradio je svoje pretrage sa modelom semantičkog pretraživanja. Slika 2. prikazuje rezultate pretrage sa Google tražilice. Kao upit je postavljen gore navedeni primjer. Prikazuju se rezultati kao adrese stranica, a isto tako rezultati semantičke pretrage informacija.



Slika 2. - Google knowledge graph

Facebook je predstavio svoju verziju semantičkog web-a nazvanu *Graph search* koja je još u testnoj fazi. Pretraga osoba na toj društvenoj mreži više neće biti ograničena na ime i prezime korisnika, odnosno podudaranje ključnih riječi, već će biti moguće upisati pitanje poput *„Tko voli filmove Woody Allena?“,* a kao rezultat dobiti popis korisnika. U isto vrijeme, na drugim stranicama će biti omogućena prijava korisnika sa Facebook identitetom. Tako se na stranici Imdb moguće prijaviti te ona preuzima podatke o korisniku. Na temelju filmova kojima je korisnik dao *„like“* algoritam preporučuje ostale. [5]

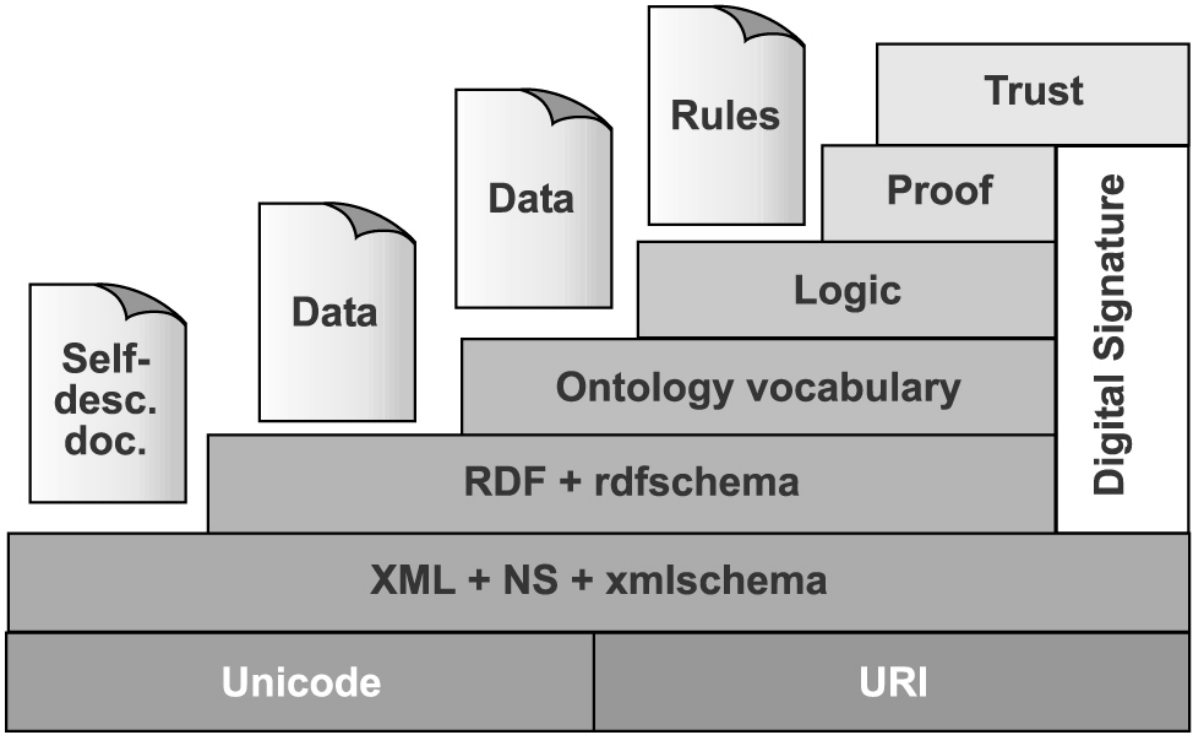
Uz stranice koje koriste semantički način pretraživanja, postoje i one koje prikupljaju znanje sa drugih stranica poput Wikipedije, s ciljem stvaranja baza podataka koje će sadržavati povezane podatke. Stranice DBpedia, Freebase, Geo Linked Data i još mnoge druge se međusobno povezuju kako bi stvorile globalni semantički web, te na globalnoj razini povezale znanje. Slika 3. prikazuje dio grafa međusobno povezanih servisa za prikupljanje i povezivanje podataka.



Slika 3. - Međusobno povezane baze podataka [6]

1.2. Slojevi semantičkog Weba

Kada je Tim Berners - Lee dao ideju za semantički web, predstavio je i model podijeljen na slojeve kao što prikazuje Slika 4., koji ima zadatak standardizirati tehnologiju semantičkog web-a. Taj model se još naziva i *„Layer Cake“*.



Slika 4. - Slojevi semantičkog web-a [7]

Ovdje je prikazan kratak opis pojedinih slojeva, a u nastavku rada će biti dan detaljniji prikaz pojedinih tehnologija semantičkog weba.

Na samome dnu je sloj *„Unicode“* i *„URI“*. Taj sloj ima zadaću jedinstveno definirati pojmove u globalnoj mreži podataka. Drugi sloj sadržan od *„XML“, „NS“* i *„XML Schema“* predstavlja temeljnu strukturu zapisa podataka. *„RDF“* je model koji samo povezuje podatke, a *„RDF Schema“* je dio koji daje osnovnu smisao i značenje vezama te omogućuje hijerarhiju. *„Ontology vocabulary“* je sloj koji nadograđuje prethodni, većim brojem semantičkih veza između podataka. *„Logic“* sloj je zadužen za razumljivo izvlačenje informacija i donošenje odluka od strane aplikacija. *„Proof“* sloj traži dokaz kako je nađen podatak točan, a *„Trust“* sloj je zadužen za provjeru da li je moguće vjerovati izvoru da su podaci točni.[8]

2. Linked Data

Ideja Linked Data odnosno povezanih podataka je temeljena na ideji semantičkog weba. Stvaranje globalne mreže znanja. Glavna značajka dosadašnjeg weba je HTML (HyperText Markup Language) i dokumenti povezani URL-om (Uniform Resource Locator) odnosno povezani samo lokacijski. U RDF (Resource Description Framework) modelu su podaci povezani RDF trojcima koji pomoću RDF Scheme, podacima daju semantiku, odnosno smisao i značenje njihove veze, te hijerarhiju.

Linked data je ustvari nadogradnja semantičkog weba kojeg je Tim Berners - Lee 2006. godine predstavio postavljajući sljedeća pravila:

- URI se koristi za definiranje resursa.
- Koriste se HTTP URI da ih ljudi mogu pronaći.
- Koriste se standardizirani formati RDF i XML.
- Koristiti što više linkova kako bi podaci bili opširniji i potpuniji.

2.1. Osnovni pojmovi povezanih podataka

2.1.1. Resurs, URI (Uniform Resource Identifier)

Resurs predstavlja osnovni entitet na webu koji može biti identificiran, imenovan, adresiran i korišten na bilo koji način.[9] Drugim riječima, resurs je bilo koji pojam iz stvarnog svijeta koji je zapisan na internetu.

URI je niz znakova koji točno definira određeni resurs na semantičkom webu. Za razliku od URL-a koji određuje mjesto, URI daje jedinstven identitet resursu. Uzmimo za primjer redatelja Woody Allena. U svijetu postoje više osoba sa tim imenom i prezimenom i pretragom na tražilicama dobivamo rezultate za više njih. Računalo ne razumije na koju osobu mislimo, odnosno za koga smo postavili upit. URI točno definira o kojoj se osobi radi npr.

- Woody Allen, filmski redatelj

  URI: http://dbpedia.org/resource/Woody_Allen

- Woody Allen, gitarist

  URI: http://dbpedia.org/resource/Allen_Woody

Ovi linkovi ujedno predstavljaju i URL, jer klikom na taj link dolazimo do stranice sa informacijama o tim osobama.

2.1.2. NS (NameSpace), Qname

Kod zapisivanja URI-a u datoteke često se pojavljuje nepreglednost zbog mnogo ponavljanja prvog dijela URI-a, odnosno domene. Kako bi izbjegli to ponavljanje i doprinijeli kompaktnosti i preglednosti zapisa u datotekama (poput XML datoteka), uvodi se NameSpace. Uzmimo za primjer gore navedeni URI:

```
http://dbpedia.org/resource/Woody_Allen
```

Kako ne bi morali svaki put ponavljati ovaj cijeli tekst, možemo ga podijeliti u ponavljani dio, domenu URI-a, odnosno NS i zamijeniti ga prefiksom, te različit dio. Dakle prefiks definiramo na sljedeći način:

```
xmlns:db = http://dbpedia.org/resource/
```

Ovime smo definirali novo proizvoljno ime *db* za naš prefiks i dodijelili smo mu NS *http://dbpedia.org/resource/*. Sada možemo u bilo kojem dijelu u nastavku dokumenta zamijeniti cijeli NS sa prefiksom i dodati mu samo nastavak kako bi imali cijeli URI.

```
db:Woody_Allen
```

Gornji izraz predstavlja Qname koji predstavlja kompaktniju verziju URI-a. U nastavku ovog rada zbog preglednosti ćemo koristiti NS.

2.2. XML (eXtensible Markup Language)

XML je standard prihvaćen od strane W3C za formatiranje podataka na način da su lako razumljivi ljudima i računalima. Podaci formatirani na taj način mogu biti procesuirani na različitim platformama, jezicima i programima.

Extensible dio u ovom jeziku govori da je proširiv. To znači da svatko može dodati svoje tag-ove. Njihov broj nije ograničen kao u HTML-u. Oni u XML-u predstavljaju *Metadata*, odnosno podatke o podacima na način da samo ime tag-a i njegov atribut postaju informacija o podatku koji se nalazi između tag-ova i koji je doslovna vrijednost.

Tag-ovi mogu:

- sadržavati doslovnu vrijednost
    - početni – tag i atribut se pišu unutar „<“ i „>“
    - završni – tag se piše unutar „</“ i „>“
    - između njih se piše doslovna vrijednost

      Primjer takvog tag-a bez atributa:

      ```
      <ime>Woody Allen</ime>
      ```

- ne sadržavati doslovnu vrijednost
    - tag i atribut se pišu između „<“ i „/ >“

      Primjer takvog taga:

      ```
      <osoba ime=“Albert Einstein“/>
      ```

XML strukturiramo na način da radimo hijerarhiju tag-ova. Postoje *root*, *child* i *parent* tag. *Root* predstavlja glavni tag koji sadrži sve ostale. Svi ostali tag-ovi u tom slučaju predstavljaju *child* tag za *root*, dok *root* predstavlja *parent* za njih. Generalno, *child* tag-ove pišemo unutar *parent* tag-a i važno je napomenuti da ih ne smijemo presijecati.

Prikazana je osnovna sintaksa jednog takvog strukturiranog zapisa.

```
<root>
  <child>
    <subchild>...</subchild>
  </child>
</root>
```

Atribute tag-ova pišemo unutar izlomljenih zagrada, a njihovu vrijednost pišemo u navodnike. Tag može sadržavati više atributa.

Komentare u XML-u pišemo unutar „<!--“ i „-->“. Tekst napisan kao komentar računalo ne procesuira.

Na početku svake XML datoteke potrebno je označiti da se radi o XML datoteci kako bi ju računalo moglo točno pročitati. To radimo na način da na početku pišemo:

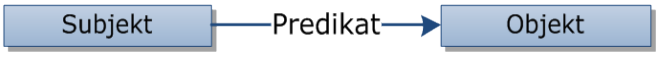
```
<?xml version=“1.0“ encoding="utf-8"?>
```

U primjeru je navedena verzija XML-a, te način enkodiranja teksta.

2.3. RDF (Resource Description Framework)

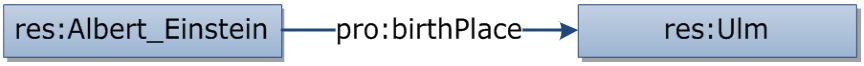
RDF je sustav za izražavanje jednostavnih izraza, odnosno veza između web resursa, koji je 1999. godine standardiziran od strane W3C-a. On je osnova na kojemu je baziran semantički web. RDF je model u kojem se vežu podaci.

Temeljna struktura povezanih podataka je RDF trojac (RDF triple) pomoću kojih je sve izraženo, a on se sastoji od subjekta, predikata i objekta. Isto kao što čovjek definira vezu između subjekta i objekta u rečenici preko predikata, tako je i u RDF trojcu definirana veza između dva web resursa (ili resursa i doslovne vrijednosti) predstavljena pomoću URI-a. Slika 5. prikazuje općeniti oblik jednog RDF trojca.



Slika 5. - Općeniti oblik RDF trojca

Uzmimo za primjer jedan RDF trojac koji daje vezu između Alberta Einsteina i njegova mjesta rođenja. Slika 6. prikazuje taj RDF trojac.



Slika 6. - Primjer RDF trojca

Svi elementi su izraženi pomoću Qname-a. Kao subjekt je dan Albert Einstein. Objekt je Ulm. Veza između njih je predikat *pro:birthPlace*. Ovaj RDF trojac pročitan od strane čovjeka bi glasio „*Albert Einstein je rođen u Ulmu*.“.

Taj RDF trojac sada razumije i računalo, jer je jednoznačno definirana osoba o kojoj se radi, grad o kojem se radi, te njihova veza. U slučaju da sada postoji više osoba s ovim imenom i više mjesta s ovim imenom, računalo bi točno znalo na koje se ovaj RDF trojac odnosi.

Objekt može biti i doslovna vrijednost. U tom slučaju je i dalje moguće pročitati podatak iz baze tako da ga čovjek razumije, ali za računalo on predstavlja samo niz znaova

2.3.1. Serijalizacija[10]

Serijalizacija je način formatiranja RDF trojaca na standardiziran način kako bi ih računala mogla čitati.

2.3.1.1. N-Triples

Najjednostavniji način zapisa RDF trojaca je *N-Triples*. U jednoj liniji se upisuju subjekt, predikat i objekt, odvojeni razmakom. Ako se kao element upisuje URI tada se on upisuje između izlomljenih zagrada „<“ i „>“, dok doslovne vrijednosti pišemo u navodnike. Na kraju svakog RDF trojca stavlja se točka. Primjer za takav format je:

```
<http://dbpedia.org/resource/Albert_Einstein>
<http://dbpedia.org/property/birthPlace> „Ulm“ .
```

2.3.1.2. Turtle (Terse RDF Triple Language)

Ovaj format je nešto kompaktniji način formatiranja. Elemente RDF trojaca, kao i u prethodnom načinu, odvajamo praznim mjestom. U ovom formatu možemo definirati prefiks zajedničkog URI-a kako ga ne bi morali ponavljati, što doprinosi kompaktnosti. Uz to možemo objediniti RDF trojce sa zajedničkim subjektom odnosno zajedničkim subjektom i predikatom, tako da ih ne moramo ponavljati.

Na sljedećem primjeru je definiran prefiks kao skraćeni dio URI-a.

```
@prefix res: <http://dbpedia.org/resource/>
@prefix pro: <http://dbpedia.org/property/>
@prefix ont: <http://dbpedia.org/ontology/>
```

Ako želimo ispisati RDF trojce sa zajedničkim subjektom tada na kraju prvog, koji sadrži zajednički subjekt, umjesto točke upisujemo točku-zarez „;“, isto kao i nakon svake sljedeće linije koja sadrži samo predikat i objekt. Nakon svih ponovljenih upisujemo točku.

```
res:Albert_Einstein  pro:birthPlace  res:Ulm ;
                     ont:field  res:Physics ;
                     ont:name  „Albert Einstein“ .
```

Ako želimo ponoviti subjekt i predikat tada na kraju prvog RDF trojca upisujemo zarez i nakon njega upisujemo objekte odvojene također zarezima. Na kraju stavljamo točku.

```
res:Albert_Einstein  ont:residence  res:Switzerland ,
                                    res:Austria-Hungary ,
                                    res:German_Empire .
```

2.3.1.3. RDF/XML

Prema W3C standardima, ova struktura je preporučena za serijalizaciju RDF grafova. Temelji se na strukturi XML formata. Kao i kod Turtle formata i u ovome se zbog kompaktnosti i neponavljanja uvodi NS. Sličnost sa XML-ovom sintaksom je ta da postoje hijerarhijski posloženi tag-ovi.

Na početku definiramo *root*, odnosno glavni tag koji će sadržavati ostale RDF trojce, a uz njega i definirati da se radi o XML datoteci.

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns:rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
                    ...
</rdf:RDF>
```

Unutar glavnog čvora upisujemo naše RDF trojce, a u atributima glavnog čvora možemo definirati prefikse NS-a.

RDF trojce strukturiramo tako što definiramo čvor koji predstavlja subjekt, sa tag-om *rdf:Description* koji označava da opisujemo nešto, a kao *rdf:about* atribut tog tag-a, upisujemo subjekt pod vrijednost atributa u navodnike. Unutar *rdf:Description* taga, dodajemo sljedeći tag koji predstavlja predikat, a on kao vrijednost atributa sadrži URI objekta ili između otvorenog i zatvorenog tag-a sadrži doslovnu vrijednost objekta.

Općenita sintaksa RDF trojaca izgleda ovako:

```
<rdf:Description rdf:about="subjekt">
      <predikat rdf:resource="objekt" />
      <predikat> doslovna vrijednost objekta </predikat>
<rdf:Description>
```

2.3.1.4. JSON (JavaScript Object Notation)

Iako je ovaj način serijalizacije nastao kao format zapisivanja podataka u JavaScript programskom jeziku, danas je jedan od najraširenijih načina formatiranja zbog svoje jednostavnosti te neovisnosti o platformama i programskim jezicima. Struktura i sami podaci su formatirani kao čisti tekstualni dokument, ali odlika ovog formata je ta što je vrlo jednostavno prebaciti podatke iz čiste tekstualne datoteke u objekte i varijable unutar programa, te ih na taj način koristiti u programskom jeziku. JSON je kao i prethodni formati lako čitljiv ljudima i računalima, te je moguće zapisati podatke hijerarhijski.

S obzirom da je JSON originalno potekao iz JavaScript-a, na primjeru možemo pokazati način formatiranja i korištenja unutar programskog jezika. Za početak definiramo jedan objekt koji će sadržavati naše podatke:

```
var JSONObject = {
        „ime“ : “Albert Einstein“,
        „područje“ : „fizika“
}
```

Ovdje smo definirali JSONObject koji sadrži polja *ime i područje*, te njihove vrijednosti *Albert Einstein* i *fizika*. Kada bi željeli doći do vrijednosti spremljene u polju *ime,* odnosno vrijednosti *Albert Einstein*, jednostavno pozovemo u programu objekt imena *JSONObject*, i nakon točke dohvatimo njegovu vrijednost u polju imena *ime* na način *JSONObject.ime*

Vrijednosti mogu poprimati oblik:

- integer (cijeli ili decimalni broj)
- string (niz znakova u navodnicima)
- boolean (*true* ili *false*)
- array (polje objekata u uglatim zagradama)
- object (objekt u vitičastim zagradama)
- null (nepostojeća vrijednost)

Ovdje je vidljivo da objekt kao svoju vrijednost, može sadržavati drugi objekt, odnosno polje drugih objekata, i upravo je to način na koji stvaramo hijerarhijsku strukturu podataka unutar ovog formata.

JSON pojedinačne objekte pišemo unutar vitičastih zagrada, a skup objekta pišemo u uglate zagrade. Ime polja i njegovu vrijednost pišemo u navodne znakove te ih odvajamo dvotočkom „:“, a parove (*ime : vrijednost*) unutar objekta odvajamo zarezom.

Na sljedećem primjeru je prikazan objekt koji u sebi sadrži polje od 3 dodatna objekta, a oni sadrže po 2 para (*ime : vrijednost*).

```
{
      objekt : [
            { „ime1“:“vrijednost“ , „ime2“:“vrijednost“ } ,
            { „ime3“:“vrijednost“ , „ime4“:“vrijednost“ } ,
            { „ime5“:“vrijednost“ , „ime6“:“vrijednost“ } ,
      ]
}
```

Kod semantičkog web-a, općeniti oblik JSON načina serijalizacije RDF trojaca odnosno subjekta, predikata i objekta je sljedeći:

```
{ „Subjekt“ : { „Predikat“ : „Objekt“  } }
```

I pomoću ovog formata zbog kompaktnosti možemo ponavljati subjekt ili subjekt i predikat, upravo na način da elemente odvajamo zarezima i stavljamo u uglate zagrade. U sljedećem primjeru je prikazan općeniti način zapisivanja RDF trojaca sa ponavljanjem subjekta.

```
{ „Subjekt“ : [ {   „Predikat1“ : „Objekt1“ ,
                    „Predikat2“ : „Objekt2“ } ]
}
```

Ovdje ponavljamo *Subjekt* i imamo 2 RDF trojca.

Subjekt – Predikat1 – Objekt1

Subjekt – Predikat2 – Objekt2

JSON-LD (JavaScript Object Notation for Linking Data)

U semantičkom web-u, kao nadogradnja osnovnog JSON načina serijalizacije, koristi se JSON-LD koji sadrži neke modifikacije namijenjene i prilagođene upravo zapisivanju povezanih podataka, uz zadržavanje sintakse osnovnog JSON-a.

Jedna od najvažnijih značajki ovog formata je uvođenje objekta *@context*. Taj dio nam omogućuje uvođenje zamjenskih imena za URI-e i definiranje njihovog tipa kako bi u samoj JSON strukturi imali jednostavniji zapis, a uz to omogućuje algoritmima točno čitanje strukture datoteke neovisno s koje stranice su podaci skinuti. Treba napomenuti da ovdje možemo uvesti zamjenu za cijeli URI ili prefiks za NS pa treba razlikovati dvotočku „:“ kojom povezujemo NS i prefiks od one koja se nalazi u sintaksi JSON-a.

Slično kao i kod CSS-a, same definicije zamjenskih imena možemo spremiti u zasebnu *jsonld* datoteku ili ih možemo zapisati direktno u datoteci sa RDF trojcima.

Uzmimo za primjer jednu datoteku koju spremimo pod ime *osoba.jsonld*. U toj datoteci vrijede pravila strukture isto kao i kod samog JSON-a.

```
{
	„@context“:
	{
		„name“ : „http://xmlns.com/foaf/0.1/name“,
		„ont“ : „http://dbpedia.org/ontology/“,
		„ont:field“ : { „@type“ : „@id“ }
	}
}
```

Ovdje smo definirali zamjensko ime *name* za cijeli URI *http://xmlns.com/foaf/0.1/name*, te prefiks NS-a *http://dbpedia.org/ontology/* kao *ont.* Nakon toga smo definirali da će Qname *ont:field* biti tip URI-a. Tada možemo u novoj JSON-LD datoteci sa našim RDF trojcima pozvati prethodno napravljenu datoteku, a ujedno i zamjenska imena.

```
{
        „@context“ : „http://putanja/do/datoteke/osoba.jsonld“,
        „@id“ : „http://dbpedia.org/resource/Albert_Einstein“,
        „name“ : „Albert Einstein“,
        „ont:field“ : „Physics“
}
```

U ovom primjeru smo pozvali prethodno navedenu datoteku, te kao subjekt preko *@id* postavili URI Alberta Einsteina. Kao jedan predikat smo postavili zamjensku riječ *name* koja predstavlja URI imena, a objekt u tom RDF trojcu je doslovna vrijednost „*Albert Einstein*“. Drugi predikat je Qname *ont:field* (područje djelovanja) kojeg smo definirali preko NS-a i prefiksa u *@context* datoteci, a kao njegov objekt smo postavili doslovnu vrijednost *Physics.*

Uz spomenute riječi *@context* (definiramo zamjenska imena tipove podataka u dokumentu ili upisujemo putanju do datoteke sa istima)*, @id* (definiramo tip vrijednosti kao URI, jednoznačno opisujemo vrijednost) i *@type* (definiramo tip vrijednosti) koje su uvedene u JSON-LD, postoje još neke koje možemo iskoristiti kako bi što bolje opisali i povezali podatke, poput *@language* (jezik na kojemu je vrijednost napisana), @value (vrijednost unaprijed definirana sa *@type*) itd...

2.3.2. RDF Store

Pod pojmom RDF Store podrazumijevamo baze podataka koje mogu pohraniti trojce. Takve baze postoje kao gotova komercijalna rješenja ili napravljena rješenja od strane samih korisnika. Najjednostavnija SQL baza sa tri kolone nam može poslužiti za pohranu trojaca, ali dolazi do problema brzine i održavanja ako uzmemo u obzir broj na koji bi količina RDF trojaca s vremenom mogla narasti. Zato postoje ciljano dizajnirana gotova rješenja koja optimiziraju bazu za pohranu i manipulaciju. Neka od njih također pružaju mogućnost automatskog skupljanja podataka sa različitih izvora i pri tome izrađuju RDF trojce.

Zbog jedinstvenog definiranja određenih resursa na webu pomoću URI-a, te baze za razliku od relacijskih baza, pružaju mogućnost objedinjenja dvaju ili više setova podataka u istu bazu, tako da ne ponavljaju upisane URI-e resursa, već ih samo povezuju. Zbog standardiziranih načina serijalizacije RDF trojaca, vrlo ih je lako prebacivati i nadopunjavati iz jedne baze u drugu, koristeći N-Triples, Turtle ili RDF/XML.[10]

2.3.3. RDFa

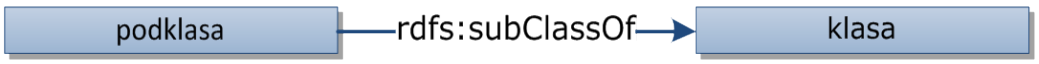
Zbog što lakšeg prikupljanja podataka navedenih programa, postavljena je ideja da u standardne HTML web stranice budu nadodani strukturirani podaci koji se spominju na istoj, ali na način da ih *parseri* mogu lako i bez greške pročitati. W3C je predstavio ideju RDFa standarda. U atribut dio HTML tag-a se upisuju podaci namijenjeni RDF ekstraktorima koji na taj način točnije i lakše izvlače podatke te stvaraju veze. Ideja je da semantički podaci budu sloj ispod samog prikaza stranice te na taj način zadržavaju željeni izgled stranice. [10]

2.4. Ontology Language

Ontologija nam služi za točno definiranje veza između resursa. Bez obzira na jezik i da li ga čita čovjek ili računalo. Osnovni RDF nam služi samo za povezivanje resursa, ali kako bi definirali semantičke odnose između njih, potrebna nam je RDF Schema.

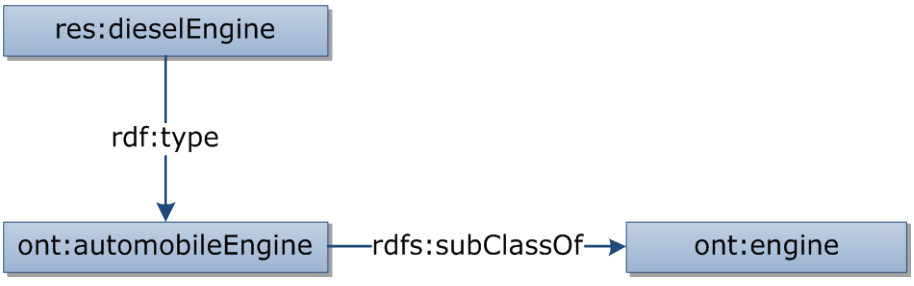
2.4.1. RDFS (RDF Schema)

RDFS je sloj iznad osnovnog RDF-a, jezik pomoću kojeg dajemo smisao vezama u RDF trojcu. RDFS stvara najosnovniju hijerarhiju i semantiku između njih. Općeniti prikaz stvaranja hijerarhije između klasa daje Slika 7.



Slika 7. - Hijerarhija RDF trojca

Uzmimo za primjer stvaranje hijerarhije između *ont:automobileEngine*, *ont:engine* i *res:dieselEngine*. Ako uzmemo da su automobilski motor i motor klase, hijerarhijski ih možemo povezati na način da je automobilski motor podklasa motora. Isto tako možemo definirati da je diesel motor vrsta automobilskog motora. Slika 7. prikazuje hijerarhijske odnose između njih.



Slika 8. - Primjer hijerarhije RDF trojaca

Ovime smo rekli da je diesel motor ujedno i motor općenito, iako nisu direktno povezani.

2.4.2. OWL (Web Ontology Language)

OWL je baziran na RDFS, te ima istu funkciju da daje smisao vezi u trojcu, odnosno da daje semantičku vezu resursima, ali uvodi više pojmova za opisivanje i ograničavanje relacija između klasa i svojstava, što omogućava računalima bolje razumijevanje veza između njih. OWL je pisan u XML-u i dio je W3C standarda od 2004. godine. Sam standard je sadržan od tri podjezika :

- OWL Lite
- OWL DL (sadrži OWL Lite)
- OWL Full (sadrži OWL DL)

koji su tim redoslijedom i kompleksniji.

Značajna nadogradnja OWL-a naspram RDFS-a je uvedena mogućnost iskazivanja jednakosti, nejednakosti i raspona između klasa i svojstava, te između URI-a. S obzirom da se pojavljuje sve više stranica koje kreiraju svoje URI-e za isti entitet, događa se da postoji više različitih URI-a na različitim domenama za jedan entitet. OWL podržava izjednačavanje tj. *owl:sameAs* definiciju te je na taj način moguće objediniti različite URI-e, a time i njihove veze u jednu globalnu vezu.

Najnoviji standard ontologijskog jezika je OWL2 koji je kompatibilan sa prošlom verzijom OWL-a i preporučen od strane W3C-a 2009. godine.

2.4.3. Ontologijski rječnici

Uz osnovne ontologijske jezike RDFS, OWL, OWL2, postoje stranice koje također na osnovu istih proširuju mogućnost opisa resursa sa proširenim vokabularom.

Možda i najznačajnija stranica što se tiče RDF trojaca, ali i ontologije je DBPedia koja sadrži ontološke riječi iz svih područja. Ostale stranice se obično orijentiraju na određeno područje pa tako postoje:

- *Dublin Core* – rječnik koji je orijentiran na opisivanje dokumenata sa opisima poput *creator, date, description, format, language, publisher, subject, title*...
- *FOAF (friend of a friend)* – namijenjen informacijama o osobama kao što su *person, organization, knows, member, name, homepage, phone*...
- *Music Ontology* – posvećen glazbi i glazbenim izdanjima sa opisima poput *album, release, label, remix, encoding, musicartist, musicgroup*...

3. SPARQL (SPARQL Protocol and RDF Query Language)

Sada kada smo objasnili osnovnu ideju semantičkog weba i povezanih podataka, te njihova obilježja i način pohrane, potrebno je te podatke i iščitati iz baze. W3C je postavio SPARQL jezik kao standard za pretragu podataka u RDF modelu.

SPARQL je za semantički web, odnosno RDF baze podataka, ono što je SQL za relacijske baze podataka. Jezik za pretragu podataka u RDF modelu, koji nije ograničen samo na baze podataka. S njim je moguće pretraživati bilo koji format zapisanih RDF trojaca, poput XML datoteka. Treba napomenuti da je u osnovnoj verziji SPARQL 1.0 moguće samo pretraživati i čitati podatke, „*read only*“, ali nije moguće upisivati nove ili raditi izmjene na postojećima u bazi.

3.1. Sintaksa

Kod upita putem SPARQL-a postoje varijable. One su namijenjene kako bi u njih spremali rezultat pretrage ili ih koristili samo privremeno kako bi filtrirali rezultate. Varijabla se označava sa upitnikom ispred imena i nije ih potrebno unaprijed deklarirati.

Slično kao i u SQL-u, upit postavljamo sa kombinacijom naredbi SELECT i WHERE.

U SELECT dijelu upisujemo varijable koje želimo pronaći, odnosno varijable koje želimo zadržati kao rezultat. Važno je napomenuti da u nastavku možemo deklarirati i dodatne varijable koje će nam pomoći u pretrazi, ali ako nisu spomenute u SELECT dijelu, one neće biti u rezultatima. Ako želimo zadržati sve varijable korištene u pretrazi, tada poput SQL-a možemo upisati zvjezdicu „*“ umjesto varijabli.

WHERE nam predstavlja dio upita u koji upisujemo trojce sa varijablama. Oni se pišu između vitičastih zagrada, „{„ i „}“, te kao u Turtle sintaksi elementi trojaca, odnosno subjekt, objekt i predikat su odvojeni praznim mjestom, a na kraju linije pišemo točku. URI-e stavljamo između izlomljenih zagrada „<“ i „>“, doslovne vrijednosti pišemo u navodnike, a Qname pišemo bez ikakvih dodatnih znakova. Također, koriste se točka-zarez „;“ za ponavljanje subjekta (sa različitim predikatom i objektom) ili „,“ za ponavljanje subjekta i predikata (sa različitim objektom). Umjesto nepoznatog elementa trojca upisujemo varijablu.

Također, slično kao i Turtle format, možemo odrediti prefikse koji će nam omogućiti kompaktniji prikaz trojaca.

Kroz sljedeće primjere su prikazani SPARQL upiti prema bazi podataka koja sadržava periodni sustav elemenata. Određen je prefiks *daml* jer je *.owl* datoteka preuzeta sa stranice http://www.daml.org/2003/01/periodictable/PeriodicTable.owl, te je postavljena u lokalnu bazu.

Za početak definiramo prefikse kako bi prikaz upita bio što kompaktniji.

```
PREFIX daml:<http://www.daml.org/2003/01/periodictable/PeriodicTable#>
```

Sada možemo postaviti jednostavan upit kako bi ispisali sve RDF trojce u bazi. Slika 9. predstavlja upit i rezultate pretrage.

```
PREFIX daml:<http://www.daml.org/2003/01/periodictable/PeriodicTable#>

SELECT * WHERE {
?subjekt ?predikat ?objekt .
}
```

subjekt	predikat	objekt
http://www.daml.org/2003/01/periodictable/PeriodicTable	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#Ontology
http://www.daml.org/2003/01/periodictable/PeriodicTable	http://www.w3.org/2002/07/owl#versionInfo	$Id: PeriodicTable.owl,v 1.10 2004/02/05 15:49:58 mdean Exp $
http://www.daml.org/2003/01/periodictable/PeriodicTable	http://www.w3.org/2000/01/rdf-schema#comment	Periodic Table of the Elements
http://www.daml.org/2003/01/periodictable/PeriodicTable#Classification	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#Class

Slika 9. - SPARQLSELECT

U ovom primjeru zvjezdicom „*“ smo naveli da želimo ispisati sve navedene varijable, odnosno subjekt, predikat i objekt.

Sada iz baze želimo preuzeti sve kemijske elemente, njihove oznake i njihove atomske brojeve. Pod SELECT dio je dodana naredba DISTINCT. Ona u slučaju ponovljenih vrijednosti u rezultatima filtrira sve jednake i ostavlja samo jednu. Slika 10. prikazuje sljedeći upit.

```
PREFIX daml:<http://www.daml.org/2003/01/periodictable/PeriodicTable#>

SELECT DISTINCT ?ime ?simbol ?atomskiBroj WHERE {
?URI daml:element ?element .
?element daml:name ?ime .
?element daml:symbol ?simbol .
?element daml:atomicNumber ?atomskiBroj .
}
```

ime	simbol	atomskiBroj
hydrogen	H	1
helium	He	2
lithium	Li	3
beryllium	Be	4
boron	B	5
carbon	C	6
nitrogen	N	7
oxygen	O	8
fluorine	F	9

Slika 10. - SPARQL DISTINCT

U SELECT dijelu smo također naveli samo varijable *?ime*, *?simbol* i *?atomskiBroj* jer ne želimo ispisivati ostale (*?URI*, *?element*) koje su nam služile samo za pretragu i sadrže URI-e. U prvom RDF trojcu upita smo postavili da želimo u varijablu *?element* spremiti sve vrijednosti koje sa *?URI* veže ontologija *daml:element* odnosno URI vrijednosti koje predstavljaju kemijske elemente. U drugom redu smo iz tih filtriranih elemenata izvukli imena, odnosno u varijablu *?ime* koja predstavlja objekt smo spremili doslovne vrijednosti imena elemenata koje su povezane preko ontologije *daml:name* sa odgovarajućim URI-jem elementa. U trećem i četvrtom redu smo na isti način izvukli simbol i atomski broj pojedinog elementa preko odgovarajuće ontologije i spremili ih u varijable *?simbol* i *?atomskiBroj*.

U sljedećem primjeru koji prikazuje Slika 11. ćemo prethodni upit zapisati kao ponovljene RDF trojce, jer se varijabla odnosno subjekt u njima ponavlja. Prvi RDF trojac u kojemu je varijabla *?element* subjekt, pišemo normalno, a na kraju umjesto točke upisujemo točka-zarez „;“. Sada u preostala dva ne moramo ponavljati subjekt, već samo pišemo predikat i objekt koje odvajamo također točkom-zarez „;“, a na kraju svih ponovljenih stavljamo točku.

```
PREFIX daml:<http://www.daml.org/2003/01/periodictable/PeriodicTable#>

SELECT DISTINCT ?ime ?simbol ?atomskiBroj WHERE {
?URI daml:element ?element .
?element daml:name ?ime ;
         daml:symbol ?simbol ;
         daml:atomicNumber ?atomskiBroj .
} ORDER BY ?atomskiBroj LIMIT 10 OFFSET 20
```

ime	simbol	atomskiBroj
scandium	Sc	21
titanium	Ti	22
vanadium	V	23
chromium	Cr	24
manganese	Mn	25
iron	Fe	26
cobalt	Co	27
nickel	Ni	28
copper	Cu	29
zinc	Zn	30

Slika 11. - SPARQL ORDER BY, LIMIT, OFFSET

Također smo proširili upit naredbama ORDER BY kojom smo poredali rezultate prema varijabli *?atomskiBroj*. Kada bi željeli poredati rezultate od najvećeg prema najmanjemu, tada pišemo ORDER BY DESC.

Naredba LIMIT ograničava broj dobivenih rezultata u ovom slučaju na 10. Vrlo korisna naredba kada pretražujemo ogromne baze podataka.

OFFSET naredbom specificiramo od kuda želimo početi ispisivati rezultate što je u ovom slučaju od dvadesetog pa nadalje.

Kako bi prikazali naredbu OPTIONAL, ovaj upit ćemo proširiti tako da iz baze dohvatimo atomske mase elemenata. Upravo iz razloga što one nisu poznate kod svih elemenata, odnosno RDF trojci nisu zapisani (nedostaje podatak), običan SQL upit ne može prikazati te rezultate, tj. doći će do greške kod varijabli jer vrijednosti ne postoje. Tada možemo naredbom OPTIONAL koju pišemo jednako kao WHERE, ali unutar WHERE-a prikazati i RDF trojce bez svih elemenata, tako da RDF trojce upišemo pod nju. Slika 12. prikazuje takav primjer.

```
PREFIX daml:<http://www.daml.org/2003/01/periodictable/PeriodicTable#>

SELECT DISTINCT ?ime ?simbol ?atomskaMasa WHERE {
?URI daml:element ?element .
?element daml:name ?ime ;
         daml:symbol ?simbol .
OPTIONAL { ?element daml:atomicWeight ?atomskaMasa . }
} ORDER BY ?atomskaMasa
```

ime	simbol	atomskaMasa
ununtrium	Uut	
ununpentium	Uup	
ununseptium	Uus	
ununhexium	Uuh	
ununoctium	Uuo	
hydrogen	H	1.00794
helium	He	4.002602
lithium	Li	6.941
beryllium	Be	9.012182
boron	B	10.811
carbon	C	12.0107
nitrogen	N	14.0067
oxygen	O	15.9994

Slika 12. - SPARQL OPTIONAL

U sljedećem primjeru koji prikazuje Slika 13. prikazana je naredba UNION.

```
PREFIX daml:<http://www.daml.org/2003/01/periodictable/PeriodicTable#>

SELECT DISTINCT ?ime ?simbol ?atomskiBroj WHERE {
?URI daml:element ?element .
?element daml:name ?ime ;
         daml:symbol ?simbol ;
         daml:atomicNumber ?atomskiBroj .
{ ?element daml:group daml:group_1 } UNION { ?element daml:group daml:group_4 } UNION { ?
element daml:group daml:group_6 }
} ORDER BY ?atomskiBroj
```

ime	simbol	atomskiBroj
hydrogen	H	1
lithium	Li	3
sodium	Na	11
potassium	K	19
rubidium	Rb	37
caesium	Cs	55
francium	Fr	87
titanium	Ti	22
zirconium	Zr	40
hafnium	Hf	72

Slika 13. - SPARQL UNION

UNION se kao i OPTIONAL piše unutar WHERE dijela i povezuje više setova RDF trojaca od kojih barem jedan set mora biti zadovoljen kako bi se smatrali za rezultate. U ovom primjeru smo na dosadašnji upit nadodali onaj koji traži elemente smještene u određenoj grupi elemenata u periodnom sustavu elemenata. Ovdje smo odredili UNION-om da može pripadati u grupu 1, 4 ili 6, odnosno da elementi pohranjeni u varijablu *?element* mora biti vezan preko predikata *daml:group* sa jednim od tri ponuđena objekta, a to su: *daml:group_1, daml:group_4* i *daml:group_6.*

Slika 14. prikazuje naredbu FILTER. Nju pišemo unutar WHERE dijela, te u zagrade upisujemo uvjete prema kojima želimo filtrirati rezultate. FILTER je moguće staviti i unutar UNION ili OPTIONAL naredbi. U danom primjeru smo dali uvjete da se ispišu svi elementi koji imaju atomski broj između 20 i 80, te imena započinju sa „c“. Opcija *„i“* određuje da se u tom filteru velika i mala slova ne uzimaju u obzir.

```
PREFIX daml:<http://www.daml.org/2003/01/periodictable/PeriodicTable#>

SELECT DISTINCT ?ime ?simbol ?atomskiBroj WHERE {
?URI daml:element ?element .
?element daml:name ?ime ;
         daml:symbol ?simbol ;
         daml:atomicNumber ?atomskiBroj .
FILTER (?atomskiBroj > 20 && ?atomskiBroj < 80 && regex(?ime, "^C", "i"))
} ORDER BY ?atomskiBroj
```

ime	simbol	atomskiBroj
chromium	Cr	24
cobalt	Co	27
copper	Cu	29
cadmium	Cd	48
caesium	Cs	55
cerium	Ce	58

Slika 14. - SPARQL FILTER

Svi ovi upiti su usmjereni na bazu podataka koja sadržava RDF trojce. SPARQL podržava i pretragu datoteka koje na neki od standardiziranih načina serijalizacije također sadrže RDF trojce. U tom slučaju ispred naredbe SELECT možemo naredbom FROM pozvati lokalnu datoteku ili datoteku na serveru. Uz naredbu FROM NAMED možemo pozvati RDF graf.

Kako bi saznali u kojem točno RDF grafu je pronađen željeni RDF trojac, možemo koristiti naredbu GRAPH koja se upisuje pod WHERE sa željenim upitom. Ona u odabranu varijablu sprema ime tog RDF grafa.

Uz SELECT pretragu, kod SPARQL jezika imamo još tipove pretraga ASK, CONSTRUCT i DESCRIBE.

ASK naredba samo provjerava da li postoji rezultat za naš upit, a kao rezultat vrača *true* ili *false*.

CONSTRUCT i DESCRIBE su vrlo slične naredbe kojima možemo (ako endpoint podržava), na temelju rezultata upita kreirati nove grafove i zapisati ih kao datoteku ili samo prikazati u nekoj od podržanih načina serijalizacije. Slika 15. prikazuje najjednostavniju sintaksu oba načina gdje je subjekt stalan u CONSTRUCT dijelu i kao takav će biti upisan. DESCRIBE dio se također može proširiti sa dijelom WHERE i ostalim prethodno navedenim naredbama za pretraživanje.

```
PREFIX daml:<http://www.daml.org/2003/01/periodictable/PeriodicTable#>

CONSTRUCT { daml:element ?p ?o }
WHERE { daml:element ?p ?o . }

DESCRIBE { daml:element }
```

Slika 15. - SPARQL CONSTRUCT, DESCRIBE

Krajem 2012. predložen je SPARQL 1.1 za standard koji sadržava još veća proširenja osnovnog SPARQL jezika, te dodaje vrlo bitne funkcije za manipuliranje RDF bazama. Uvode se INSERT i DELETE, osnovne naredbe za umetanje, brisanje i izmjenu podataka u bazi, te na taj način SPARQL postaje *Read / Write* jezik. Iako još nije službeno prihvaćen kao standard, mnogo komercijalnih, ali i besplatnih programa namijenjenih korištenju povezanih podataka i semantičkom web-u ga već koriste.

Kako bi prikazali INSERT naredbu koja upisuje RDF trojce, ispraznit ćemo bazu. Slika 16. prikazuje sintaksu INSERT INTO naredbe i sam RDF trojac koji je sastavljen od subjekta zapisanog pomoću Qname URI-a. Predikat je također Qname URI-a koji govori da se radi o imenu. Objekt je doslovna vrijednosti imena. U izlomljene zagrade se upisuje RDF Graph u koji postavljamo, a u ovom slučaju je to lokalni zadani pa ostavljamo prazno.

```
PREFIX daml:<http://www.daml.org/2003/01/periodictable/PeriodicTable#>

INSERT INTO <> { daml:Fe daml:name "Željezo" }
```

Ubačeno trojaca: 1

Slika 16. - SPARQL INSERT INTO

Slika 17. prikazuje DELETE FROM sintaksu koja kao i INSERT u izlomljenim zagradama može sadržavati RDF graf. Unutar vitičastih zagrada se upisuju RDF trojci prema kojima se vrši brisanje. Ako bi naveli sva tri elementa kao varijable, cijela baza bi bila obrisana.

```
PREFIX daml:<http://www.daml.org/2003/01/periodictable/PeriodicTable#>

DELETE FROM <> { ?subjekt ?objekt "Željezo" }
```

Obrisano trojaca: 1

Slika 17. - SPARQL DELETE FROM

3.2. SPARQL Endpoint

Predstavlja protokol kojim možemo slati upite prema otvorenim RDF bazama. Osim što nam olakšava pretraživanje drugih baza iz vlastitih aplikacija, većinom na tom URI-u postoje i HTML forme preko kojih direktno možemo pretraživati tu bazu i koje automatski prikazuju rezultate. Najčešće su rezultati prikazani kao tablica, ali postoji mogućnost i prikaza rezultata putem navedenih načina serijalizacije. Tako na primjer DBpedia je postavila Virtuoso endpoint na linku http://dbpedia.org/sparql preko kojeg možemo pretraživati njenu bazu.

4. Povezani podaci u praksi pomoću PHPa i MySQLa

Ovdje će pomoću PHP-a biti izrađena stranica iza koje će stajati MySQL baza podataka koja će nam služiti kao RDF Store, za pohranu RDF trojaca. Njih ćemo moći ubacivati u bazu iz datoteka koje ih sadrže poput *.rdf, .xml i .owl*, te ostalih koje zadovoljavaju jedan od formata standardnih načina serijalizacije kao što su RDF/XML, N-Triples i Turtle. Editiranje samih RDF trojaca će biti moguće preko SPARQL 1.1 upita, te prikaz cijele ili djelomične baze na samoj stranici u tablici.

Najjednostavniji način za upisivanje RDF trojaca u MySQL bazu podataka bi bio da napravimo tablicu sa tri stupca, a zatim popunjavamo redove sa RDF trojcima, odnosno prvi stupac bi bio subjekt, drugi predikat, a treći objekt.

Takav način bi bio funkcionalan za male baze, odnosno primjere. Kada bi željeli napraviti ogromnu bazu sa milijardama RDF trojaca, takav način zapisa baze ne bi funkcionirao zbog brzine traženja podataka, veličine i mogućnosti održavanja iste.

Zato ćemo u ovom radu koristiti ARC2[15].

4.1. ARC2

Kombinacijom ovog sustava, PHP-a i MySQL-a možemo stvoriti vrlo kvalitetne semantičke aplikacije i RDF baze podataka. ARC2 je kompletno rješenje koje sadrži *parsere* i serijalizatore (RDF/XML, N-Triples, Turtle...). Zatim koristi MySQL bazu podataka kao RDF Store. Koristi i SPARQL 1.1 endpoint. Pomoću njega možemo pretraživati HTML stranice s podacima u mikroformatima i spremati ih u našu bazu. Podržava i naknadna različita proširenja.

Uz sve to je besplatan, ali nažalost 2011. je prestao sa razvojem. Unatoč tome sve njegove funkcije su usklađene sa zadnjim W3C standardima.

4.2. Ostali programi

Uz ARC2, u izradi stranice i baze podataka su korišteni još Adobe Dreamweaver CS6, WampServer (koji uključuje PHP 5.4.3, MySQL 5.5.24 i Apache 2.2.22), Corel PaintShop Pro X4 i DBpedia kao izvor *.rdf* datoteka koje će sadržavati RDF trojce. Te datoteke su skraćene i nešto modificirane zbog ogromnog broja RDF trojaca, ali u bazu je moguće učitati bilo koju datoteku sa interneta pod uvjetom da koristi standardan format. Datoteke se nalaze u *RDF_files* direktoriju stranice.

4.3. Instalacija ARC2 i MySQL baze podataka

Prije svega potrebno je instalirati WampServer koji će nam omogućiti korištenje PHP-a i MySQL-a. ARC2 zahtjeva server koji sadrži minimalno PHP4.3 i MySQL5.0. *Root* direktorij WampServera neka ostane *www* kao što ćemo ga koristiti u nastavku.

4.3.1. Instalacija ARC2

Na stranici https://github.com/semsol/arc2 je moguće preuzeti sve potrebne datoteke ARC2 sustava a to su *„arc2-starter-pack“* i *„arc2-master“.* Cijeli direktorij *„arc2-starter-pack“* je potrebno postaviti u *root* direktorij naše stranice, odnosno:

```
www/arc2-starter-pack
```

a datoteke iz *„arc2-master“* arhive treba postaviti u:

```
www/arc2-starter-pack/arc
```

4.3.2. Instalacija MySQL baze

Pomoću *phpMyAdmin* koji je dio WampServera kreirat ćemo novu bazu podataka imena *linkeddata.* Također kreiramo novog korisnika imena *mcfsb* i lozinke *mc* koji će imati pristup toj bazi.

Nakon toga u datoteci *config.php* koja se nalazi u *www/arc2-starter-pack,* potrebno je izmijeniti podatke o bazi i korisniku koje smo prethodno kreirali, tako da će sada dio te datoteke izgledati ovako:

```
'db_host' => 'localhost',
'db_user' => 'mc_fsb',
'db_pwd' => 'mc',
'db_name' => 'linkeddata',
```

Naravno, ako radimo stranicu za lokalno računalo tada *db_host* ostaje *localhost.* U ovoj datoteci još možemo promijeniti imena tablica u bazi, dozvoljene naredbe SPARQL-a, te dodatne opcije limita rezultata, vremena pretraživanja i drugih.

Kreiranje potrebnih tablica u bazi je automatsko te je samo potrebno prikazati stranicu *www/arc2-starter-pack.*

Ako nismo mijenjali početno ime tablica, koje je *sandbox*, tada naša *linkeddata* baza ima sljedeće kreirane tablice:

- sandbox_g2t
- sandbox_id2val
- sandbox_o2val
- sandbox_s2val
- sandbox_setting
- sandbox_triple

koje je kreirao ARC2.

4.4. Izrada stranice

Ovdje će biti prikazani dijelovi koda koji su zaduženi za upisivanje u bazu i prikaz RDF trojaca. Sam HTML i CSS je izostavljen, a pošto je PHP kod razlomljen u dijelove zbog lakšeg opisa, postoji mogućnost da se vitičaste zagrade ne podudaraju u pojedinim dijelovima. Slika 18. prikazuje *Naslovnu* stranicu.

Slika 18. - Naslovna stranica

RDF/XML datoteka.

Ovaj kompletan dio se sastoji od forme za odabir datoteke koja sadrži RDF trojce i njihovo upisivanje u bazu. Datoteka mora biti ispravna, prema standardiziranim formatima zapisa (RDF/XML, Turtle, N-Triples).

Dio koji je zadužen za učitavanje datoteke, provjeru njene ekstenzije *(.rdf, .xml i .owl)* i veličine izgleda ovako:

```
$allowedExts = array("rdf", "xml", "owl");
$tmpext = explode(".", $_FILES["file"]["name"]);
$extension = end($tmpext);
if (($_FILES["file"]["size"] < 1000000)
&& in_array($extension, $allowedExts))
{ if ($_FILES["file"]["error"] < 1)
{ if (file_exists("../upload/" . $_FILES["file"]["name"]))
{ unlink("../upload/" . $_FILES["file"]["name"]); }
move_uploaded_file($_FILES["file"]["tmp_name"], "../upload/db.rdf"); }
```

Datoteka se sprema u direktorij *www/upload/* pod imenom *db.rdf.*

Dio koji je zadužen za pozivanje ARC2 sustava i definiranje podataka za spajanje na bazu poput korisničkog imena, lozinke i imena baze izgleda ovako:

```
include_once("../arc2-starter-pack/arc/ARC2.php");
$config = array(
'db_name' => 'linkeddata',
'db_user' => 'mcfsb',
'db_pwd' => 'mc',
'store_name' => 'sandbox',
'max_errors' => 100,
);
```

Skripta je namještena tako da dozvoljava do 100 pogrešaka u sintaksi, te pogrešno zapisani trojci neće biti upisani u bazu. Postoji mogućnost promjene limita pogrešaka u kodu, u slučaju da se radi o ogromnim datotekama.

Sljedeći dio provjerava da li postoje tablice u bazi. Ako ne postoje, ARC2 ih stvara.

```
$store = ARC2::getStore($config);
if (!$store->isSetUp()) {
$store->setUp(); }
```

Ovaj dio izvršava upit nad bazom, odnosno upisuje RDF trojce u bazu iz datoteke *db.rdf* i ako je uspješan upis prikazuje popis RDF Trojaca. U suprotnom ispisuje grešku.

```
if ($store->query('LOAD <../upload/db.rdf>')) {
unlink("../upload/db.rdf");
header("Location: ../rdf_triples.php");
} else {
 echo "Došlo je do greške sa datotekom.";
}} else {
echo "Došlo je do greške sa datotekom."; }
```

Slika 19. prikazuje izgled stranice *RDF/XML Datoteka*.

Slika 19. - RDF/XML Datoteka

RDF Trojci

Ova stranica čita cijelu bazu i prikazuje RDF trojce koji su zapisani u njoj kao subjekt, objekt i predikat u tablici. Uz to postoji gumb koji briše sve RDF trojce iz baze. PHP kod izgleda ovako:

Prvi dio koda poziva ARC2 sustav, te kao u prethodnom dijelu provjerava da li postoji baza i tablice, a ako ne postoje, kreira ih.

```
include_once("arc2-starter-pack/arc/ARC2.php");
include_once('arc2-starter-pack/config.php');
$store = ARC2::getStore($arc_config);
if (!$store->isSetUp()) {
$store->setUp();
}
```

Sljedeći dio koda sprema SPARQL upit u varijablu *$q*. Pošto će na ovoj stranici biti ispisani svi RDF trojci iz baze, SPARQL upitom možemo dohvatiti sve elemente RDF trojaca odnosno subjekt, objekt i predikat i spremiti ih u varijable *?s, ?p i ?o*.

```
$q ='SELECT * WHERE { ?s ?p ?o }';
```

Nakon toga pretražujemo bazu sa upitom spremljenim u varijablu *$q* i podatke spremamo kao redove u varijablu *$rows.*

```
$r = '';
if ($rows = $store->query($q, 'rows')) {
```

Još nam preostaje iz varijable *$rows* pročitati red po red i ispisati subjekt, objekt i predikat, te ih uz strukturu zaglavlja tablice spremiti u varijablu *$r.*

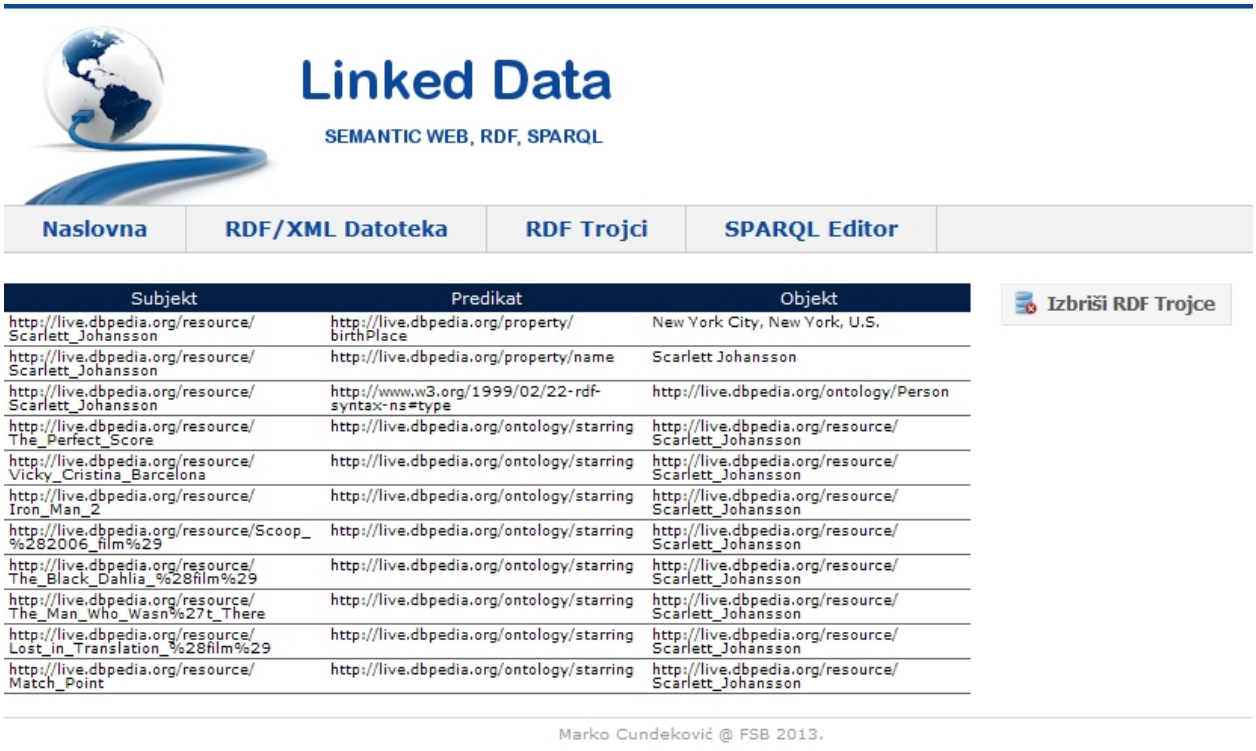
```
$r = '<table>
<th>Subjekt</th>
<th>Predikat</th>
<th>Objekt</th>'."\n";
```

```
foreach ($rows as $row) {
$r .= '
<tr><td>'.$row['s'] .'</td><td>'.$row['p'] .'</td><td>'.$row['o'] .
'</td></tr>'."\n"; }
$r .='
</table>'."\n";
```

Na kraju, ako nije postojao ni jedan RDF trojac u bazi, varijabli *$r* dodjeljujemo vrijednost *„Baza podataka je prazna!“*. U suprotnome ako smo dohvatili bar jedan RDF trojac, tada ispisujemo tu tablicu.

```
} else {
$r = 'Baza podataka je prazna!'; }
echo $r;
```

Slika 20. prikazuje stranicu *RDF Trojci* i RDF trojce koji su prethodno upisani u bazu iz jedne od datoteka koje smo spomenuli, koje su skinute sa DBpedie i malo skraćene.



Subjekt	Predikat	Objekt
http://live.dbpedia.org/resource/Scarlett_Johansson	http://live.dbpedia.org/property/birthPlace	New York City, New York, U.S.
http://live.dbpedia.org/resource/Scarlett_Johansson	http://live.dbpedia.org/property/name	Scarlett Johansson
http://live.dbpedia.org/resource/Scarlett_Johansson	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://live.dbpedia.org/ontology/Person
http://live.dbpedia.org/resource/The_Perfect_Score	http://live.dbpedia.org/ontology/starring	http://live.dbpedia.org/resource/Scarlett_Johansson
http://live.dbpedia.org/resource/Vicky_Cristina_Barcelona	http://live.dbpedia.org/ontology/starring	http://live.dbpedia.org/resource/Scarlett_Johansson
http://live.dbpedia.org/resource/Iron_Man_2	http://live.dbpedia.org/ontology/starring	http://live.dbpedia.org/resource/Scarlett_Johansson
http://live.dbpedia.org/resource/Scoop_%282006_film%29	http://live.dbpedia.org/ontology/starring	http://live.dbpedia.org/resource/Scarlett_Johansson
http://live.dbpedia.org/resource/The_Black_Dahlia_%28film%29	http://live.dbpedia.org/ontology/starring	http://live.dbpedia.org/resource/Scarlett_Johansson
http://live.dbpedia.org/resource/The_Man_Who_Wasn%27t_There	http://live.dbpedia.org/ontology/starring	http://live.dbpedia.org/resource/Scarlett_Johansson
http://live.dbpedia.org/resource/Lost_in_Translation_%28film%29	http://live.dbpedia.org/ontology/starring	http://live.dbpedia.org/resource/Scarlett_Johansson
http://live.dbpedia.org/resource/Match_Point	http://live.dbpedia.org/ontology/starring	http://live.dbpedia.org/resource/Scarlett_Johansson

Slika 20. - RDF Trojci

SPARQL Editor

U ovom dijelu stranice postoji mogućnost slanja upita na lokalnu bazu sa SPARQL 1.1 verzijom, tako da je moguće upisivati i raditi izmjene na bazi pomoću naredbi INSERT i DELETE, za razliku od osnovnog SPARQL 1.0 u kojem je moguće samo čitanje. Uz lokalnu bazu, moguće je pretraživati i internet baze kao što su DBpedia i Linkedmdb. Na njima nije moguće upisivanje i brisanje po bazi jer su zaštićene. Ovdje ćemo spomenuti skriptu koja šalje upit, te prikazuje rezultate pomoću tablice.

U prvom dijelu koda pozivamo ARC2 i provjeravamo da li se spajamo na lokalnu bazu ili jednu od ponuđenih internet baza. U slučaju lokalne, provjeravamo da li baza i tablice postoje, a u slučaju da ne postoje, ARC2 ih kreira.

```
if ($_POST['db'] == 'dbpedia') {
$config = array('remote_store_endpoint' =>
'http://dbpedia.org/sparql',);
$store = ARC2::getRemoteStore($config);
} elseif ($_POST['db'] == 'linkedmdb') {
$config = array('remote_store_endpoint' =>
'http://data.linkedmdb.org/sparql',);
$store = ARC2::getRemoteStore($config);
} else {
$store = ARC2::getStore($arc_config);
if (!$store->isSetUp()) {
$store->setUp(); } }
```

U sljedećem dijelu u varijablu *$q* spremamo SPARQL upit iz HTML forme.

```
$q = $_POST['query'];
```

Nakon toga šaljemo upit i rezultate spremamo u varijablu *$rows.* Provjeravamo koji tip upita je postavljen.

```
$html = '';
$rs = $store->query($q);
if (!$store->getErrors()) {
$type = $rs['query_type'];
```

U slučaju SELECT-a provjeravamo imena varijabli i njihove vrijednosti. Prema imenima varijabli stvaramo zaglavlje tablice, a sve vrijednosti varijabli, odnosno elemente RDF trojaca spremamo kao redove tablice u varijablu *$html.*

```
if ($type == 'select') {
$vars = $rs['result']['variables'];
$data = $rs['result']['rows'];
$html = '<br><br><table><tr>';
foreach ($vars as $var) {
$html .= '<th>'.$var.'</th>';  }
$html .= '</tr>'."\n";
foreach ($data as $row) {
$html .= '<tr>';
foreach ($row as $key => $value) {
if (!strpos($key, ' ')) {
$html .= '<td>'.$value.'</td>'; } }
$html .= '</tr>'."\n"; }
$html .= "\n".'</table>'."\n";
```

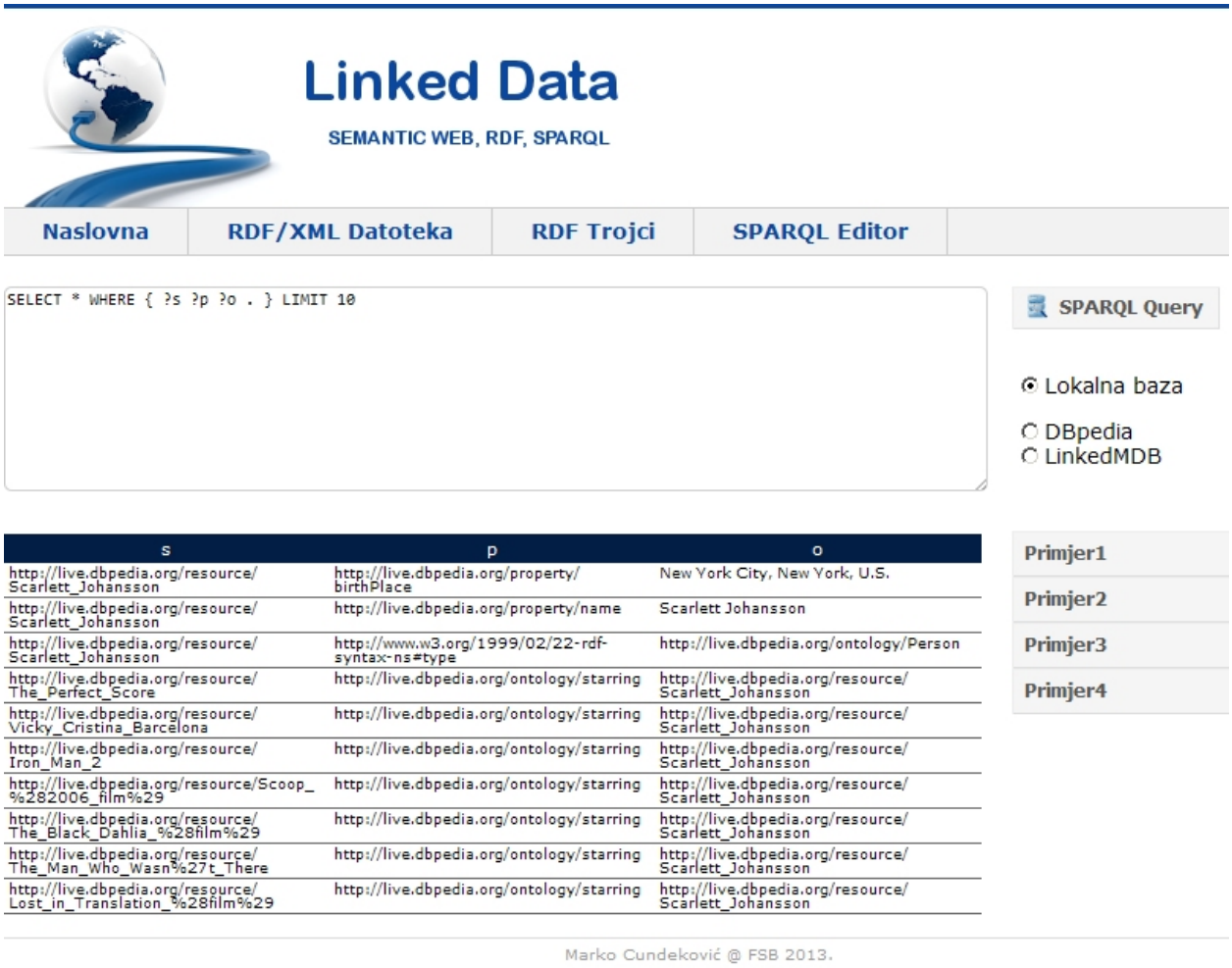
Ako je postavljen INSERT upit, tada naš rezultat u polju sadrži broj ubačenih RDF trojaca, a ako je postavljen DELETE upit tada sadrži broj obrisanih RDF trojaca. Taj broj upisujemo kao poruku u varijablu *$html*.

```
} elseif ($type == 'insert') {
$html = 'Ubačeno trojaca: '.$rs['result']['t_count'];
} elseif ($type == 'delete') {
$html = 'Obrisano trojaca: '.$rs['result']['t_count'];
}
```

Ako postoje vraćeni RDF trojci, odnosno ako je upit prošao u redu, tada ih ispisujemo uključujući i tablicu spremljenu u varijablu *$html*. U suprotnom u varijablu *$html* upisujemo *„Došlo je do greške pri upitu!“* i ispisujemo ju.

```
} else {
$html = 'Došlo je do greške pri upitu!';
} echo '</br>'.$html; }
```

Slika 21. prikazuje stranicu *SPARQL Editor* sa osnovnim SPARQL upitom i rezultatima pretrage. Sa desne strane postoji mogućnost odabira baze koju pretražujemo (lokalna baza ili internet baza). Također su dani neki primjeri upita.



s	p	o
http://live.dbpedia.org/resource/Scarlett_Johansson	http://live.dbpedia.org/property/birthPlace	New York City, New York, U.S.
http://live.dbpedia.org/resource/Scarlett_Johansson	http://live.dbpedia.org/property/name	Scarlett Johansson
http://live.dbpedia.org/resource/Scarlett_Johansson	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://live.dbpedia.org/ontology/Person
http://live.dbpedia.org/resource/The_Perfect_Score	http://live.dbpedia.org/ontology/starring	http://live.dbpedia.org/resource/Scarlett_Johansson
http://live.dbpedia.org/resource/Vicky_Cristina_Barcelona	http://live.dbpedia.org/ontology/starring	http://live.dbpedia.org/resource/Scarlett_Johansson
http://live.dbpedia.org/resource/Iron_Man_2	http://live.dbpedia.org/ontology/starring	http://live.dbpedia.org/resource/Scarlett_Johansson
http://live.dbpedia.org/resource/Scoop_%282006_film%29	http://live.dbpedia.org/ontology/starring	http://live.dbpedia.org/resource/Scarlett_Johansson
http://live.dbpedia.org/resource/The_Black_Dahlia_%28film%29	http://live.dbpedia.org/ontology/starring	http://live.dbpedia.org/resource/Scarlett_Johansson
http://live.dbpedia.org/resource/The_Man_Who_Wasn%27t_There	http://live.dbpedia.org/ontology/starring	http://live.dbpedia.org/resource/Scarlett_Johansson
http://live.dbpedia.org/resource/Lost_in_Translation_%28film%29	http://live.dbpedia.org/ontology/starring	http://live.dbpedia.org/resource/Scarlett_Johansson

Slika 21. - SPARQL Editor

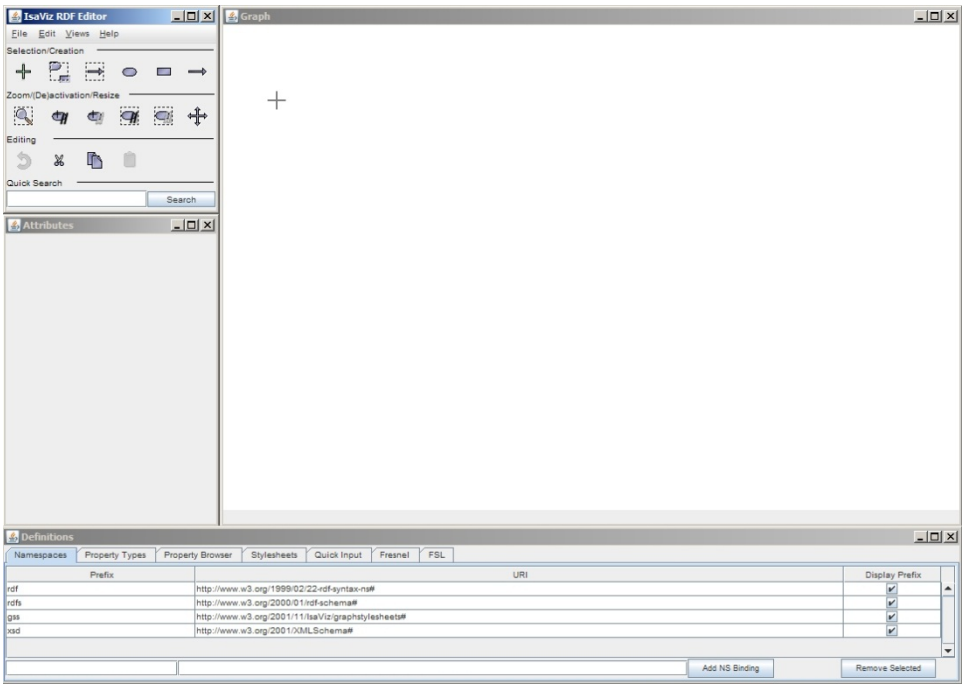
5. IsaViz 3.0

IsaViz 3.0 je program sa grafičkim sučeljem pomoću kojega možemo prikazati RDF trojce i njihove veze iz datoteka, a isto tako možemo i stvarati nove setove podataka. Ovaj program je rađen u Javi i zadnja verzija je IsaViz 3.0 koja je izašla 2007. godine. Za funkcioniranje ovog programa, potrebna su još dodatna dva, a to su GraphViz i Java. Sva tri programa su potpuno besplatna za korištenje. Nakon instalacije GraphViz-a i Jave, program IsaViz pokrećemo sa *run.bat.*

Korisničko sučelje se sastoji od osnovna 4 prozora, a to su:

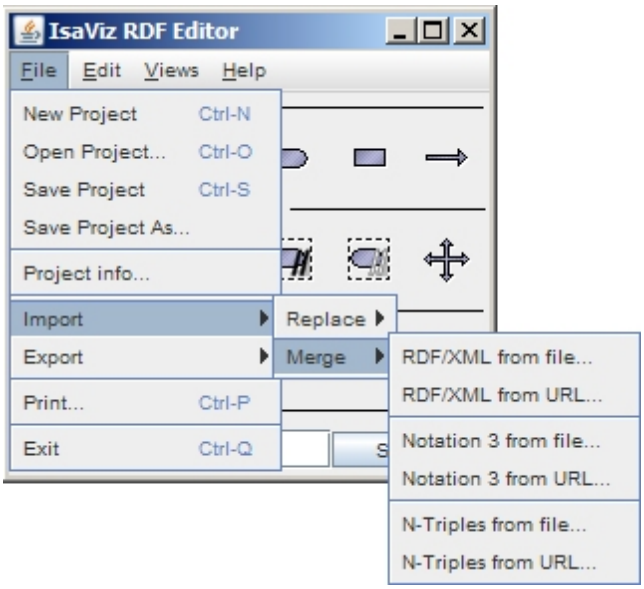
- IsaViz RDF Editor – prozor u kojem se nalaze svi glavni izbornici, te su postavljeni alati za selektiranje i kreiranje RDF trojaca, te alati za zumiranje,
- Graph – prozor u kojem se grafički prikazuju RDF trojci i njihove veze,
- Attributes – prozor koji prikazuje atribute selektiranog elementa RDF trojca,
- Definitions – prozor u kojem možemo kreirati i editirati NS, URI, prefikse, imena...

Slika 22. prikazuje osnovno sučelje nakon pokretanja programa.



Slika 22. - Korisničko sučelje IsaViz 3.0

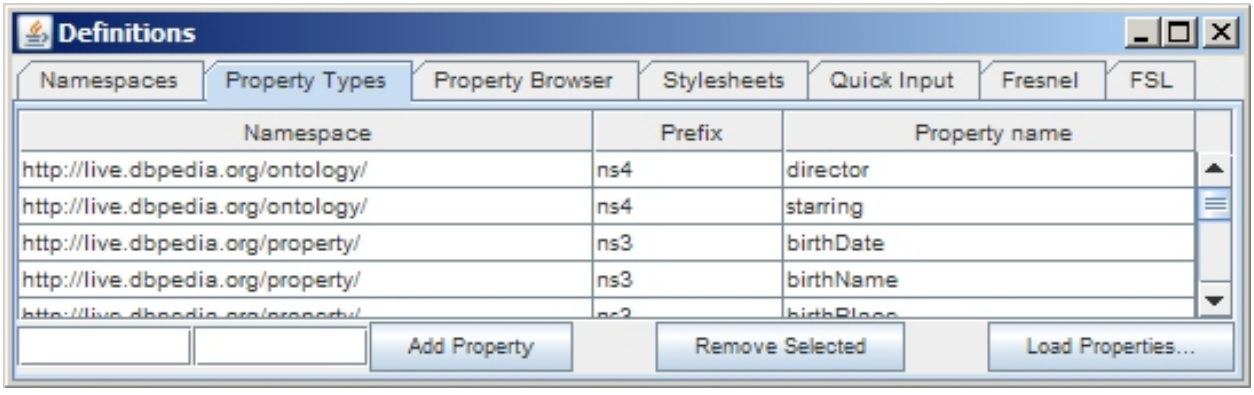
Slika 23. prikazuje glavni izbornik u IsaViz 3.0 RDF Editor prozoru.



Slika 23. - IsaViz 3.0 glavni izbornik

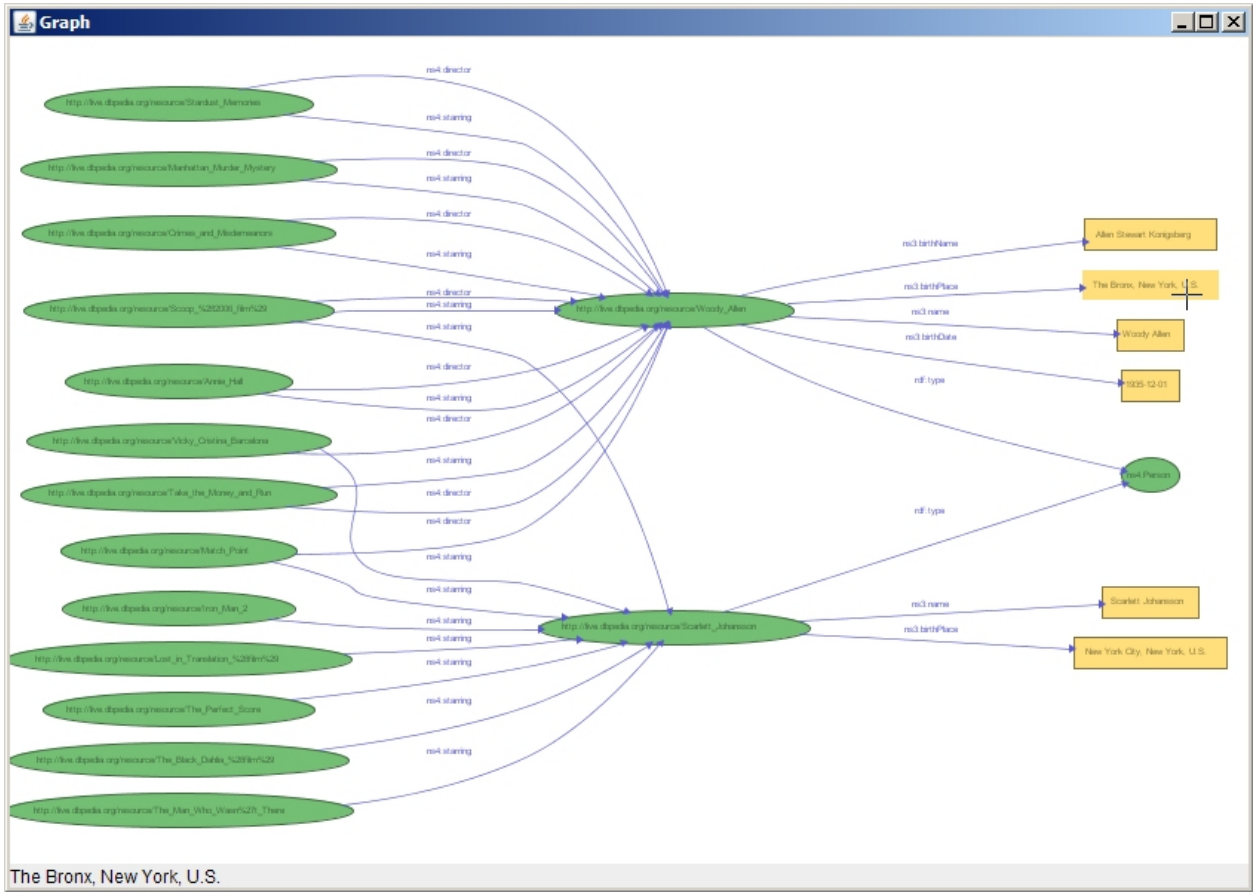
U glavnom izborniku imamo standardne opcije spremanja, otvaranja i novih projekata. U njemu dodajemo datoteke koje sadrže RDF trojce. Postoje opcije *Replace* i *Merge.* Ako želimo više setova podataka objediniti u jedan, tada odabiremo *Merge* i format naše datoteke, te je otvaramo. Također možemo dodavati RDF trojce direktno preko URL-a. Ovdje postoji i opcija *Export* kojom možemo RDF trojce koje smo učitali, editirali ili stvorili, vrlo lako spremiti u datoteku u jednom od ponuđenih formata.

U *Definitions* prozoru su ispisani svi NS, prefiksi i sufiksi koje sadrže datoteke koje smo učitali. Slika 24. prikazuje *Definitions* prozor.



Slika 24. - IsaViz 3.0 Definitions prozor

Slika 25. prikazuje *Graph* prozor i u njemu graf naših RDF trojaca iz dvije datoteke.



Slika 25. - IsaViz 3.0 Graph prozor

Zelenom bojom su ispunjeni resursi u kojima su upisani njihovi URI. Žutom bojom u pravokutnicima se nalaze doslovne vrijednosti. Plavom bojom su prikazane veze, odnosno predikati. Prema strelicama je moguće prepoznati što je subjekt, a što objekt.

U ovom prozoru možemo grafički kreirati nove RDF trojce ili mijenjati postojeće. Pošto je ovaj graf stvoren iz dvije datoteke, on je automatski njih povezao prema URI-u. U izbornicima postoji opcija da sada taj novonastali set RDF trojaca spremimo kao jednu datoteku sa novim vezama. Program pruža mogućnost provjere sintakse, te javlja greške u strukturi.

6. Zaključak

Iako je semantički web i povezani podaci kao ideja već dugo poznata, tek zadnjih godina se pokazala kao stvarna i nezaobilazna potreba. Zbog širenja interneta i rasta broja informacija na njemu, sve teže je filtrirati one potrebne. Osim toga, proizvode se sve više uređaja čija se funkcionalnost oslanja na internet poput pametnih telefona, tableta, prijenosnih računala, autiju, pa čak i hladnjaka. U cijeloj toj mreži, brzina dolaženja do podataka i njihova kvaliteta postaje vrlo bitan faktor.

Povezani podaci osim što ljudima olakšavaju upotrebu interneta, također imaju vrlo kvalitetne temelje za algoritme i strojeve koji koriste umjetnu inteligenciju. Upravo semantika kod povezanih podataka olakšava algoritmima kod kvalitetnog zaključivanja.

Semantički web ne mijenjaju strukturu današnjeg interneta, već on upravo postaje struktura interneta. U Web 3.0 standardu i dalje će se koristiti standardne web tehnologije, ali internet svojim širenjem danas ostavlja mnogo praznog i neiskorištenog mjesta, koje će povezani podaci ispuniti dodajući smisao između tog velikog broja nepovezanih podataka.

7. LITERATURA

[1] Tim Berners-Lee: The next Web of open, linked data; http://www.youtube.com/watch?v=OM6XIICm_qo, 2013.

[2] Dean Allemang, James Hendler: Semantic Web for the Working Ontologist, Second Edition 2011.

[3] http://solutions.wolterskluwer.com/blog/wp-content/uploads/webtimeline.jpg, 2013.

[4] http://hr.wikipedia.org/wiki/Semantika, 2013.

[5] http://www.theverge.com/2013/1/15/3878950/facebook-announces-graph-search, 2013.

[6] http://www.inforbix.com/wp-content/uploads/2011/05/Picture-49.png, 2013.

[7] http://www.emeraldinsight.com/content_images/fig/0150260403001.png, 2013.

[8] Grigoris Antoniou, Frank van Harmelen: A Semantic Web Primer, Second Edition, 2008.

[9] http://en.wikipedia.org/wiki/Resource_(Web), 2013.

[10] John Domingue, Dieter Fensel, James A. Hendler: Handbook of Semantic Web Technologies, 2011.

[11] Axel Polleres: Semantic Web Technologies, From Theory to Standards, 2010.

[12] Vijayan Sugumaran, Jon Atle Gulla: Applied Semantic Web Technologies, 2011.

[13] Bob DuCharme: Learning SPARQL, 2011.

[14] Toby Segaran, Colin Evans, Jamie Taylor: Programming the Semantic Web, 2009.

[15] Akilah McIntyre, E. E. Durham: ARC RDF Store, 2011.

8. PRILOZI

I. CD-R disc

II. Web stranica (.zip file)